VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
FACULTY OF INFORMATION TECHNOLOGY

ÚSTAV POČÍTAČOVÉ GRAFIKY A MULTIMÉDIÍ
DEPARTMENT OF COMPUTER GRAPHICS AND MULTIMEDIA

DESKOVÁ HRA V ROZŠÍŘENÉ REALITĚ NA DOTYKOVÉM STOLE

AR-BASED BOARD GAME ON TOUCH-ENABLED TABLE

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE — **ANETA HELEŠICOVÁ**
AUTHOR

VEDOUCÍ PRÁCE — **Ing. MICHAL KAPINUS**
SUPERVISOR

BRNO 2018

Zadání bakalářské práce/21273/2017/xheles02

Vysoké učení technické v Brně - Fakulta informačních technologií

Ústav počítačové grafiky a multimédií Akademický rok 2017/2018

Zadání bakalářské práce

Řešitel: **Helešicová Aneta**
Obor: Informační technologie
Téma: **Desková hra v rozšířené realitě na dotykovém stole**
AR-Based Board Game on Touch-Enabled Table
Kategorie: Uživatelská rozhraní

Pokyny:

1. Prostudujte koncept rozšířené reality a její využití při návrhu uživatelských rozhraní.
2. Seznamte se s fakultní experimentální platformou ARTable, jejími možnostmi a aplikačním rozhraním.
3. Vyberte vhodné metody a nástroje a navrhněte uživatelské rozhraní pro deskovou hru, které bude promítané na plochu dotykového stolu, jenž umožní ovládání této hry.
4. Navržené rozhraní implementujte a integrujte do systému ARTable.
5. Proveďte experimenty, demonstrujte a diskutujte vlastnosti vašeho řešení.
6. Vytvořte stručný plakát nebo video prezentující vaši bakalářskou práci, její cíle a výsledky.

Literatura:

- Dle pokynů vedoucího.

Pro udělení zápočtu za první semestr je požadováno:

- Splnění prvních tří bodů zadání.

Podrobné závazné pokyny pro vypracování bakalářské práce naleznete na adrese http://www.fit.vutbr.cz/info/szz/

Technická zpráva bakalářské práce musí obsahovat formulaci cíle, charakteristiku současného stavu, teoretická a odborná východiska řešených problémů a specifikaci etap (20 až 30% celkového rozsahu technické zprávy).

Student odevzdá v jednom výtisku technickou zprávu a v elektronické podobě zdrojový text technické zprávy, úplnou programovou dokumentaci a zdrojové texty programů. Informace v elektronické podobě budou uloženy na standardním nepřepisovatelném paměťovém médiu (CD-R, DVD-R, apod.), které bude vloženo do písemné zprávy tak, aby nemohlo dojít k jeho ztrátě při běžné manipulaci.

Vedoucí: **Kapinus Michal, Ing.**, UPGM FIT VUT
Datum zadání: 1. listopadu 2017
Datum odevzdání: 16. května 2018

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
Fakulta informačních technologií
Ústav počítačové grafiky a multimédií
612 66 Brno, Božetěchova 2

L.S.

doc. Dr. Ing. Jan Černocký
vedoucí ústavu

Abstrakt

Práce se zabývá využitím interaktivního stolu ARTable pro hraní komplikované deskové hry. Jejím výsledkem je promítané uživatelské rozhraní pro hru World of Warcraft Desková hra s ovládáním skrze dotykovou plochu v jazyce C++.

Abstract

In the course of this Bachelor thesis, an application using interactive table ARTable for playing complicated board games will be created. The output of this thesis is projected user interface for World of Warcraft Board game controlled by touch-enabled surface in C++ programming language.

Klíčová slova

Rozšířená realita, Robot Operating System, interaktivní stůl, ARTable, hra.

Keywords

Augmented reality, Robot Operating System, interactive table, ARTable, game.

Citace

HELEŠICOVÁ, Aneta. *Desková hra v rozšířené realitě na dotykovém stole*. Brno, 2018. Bakalářská práce. Vysoké učení technické v Brně, Fakulta informačních technologií. Vedoucí práce Ing. Michal Kapinus

Desková hra v rozšířené realitě na dotykovém stole

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracovala samostatně pod vedením pana Ing. Michala Kapinuse. Uvedla jsem všechny literární prameny a publikace, ze kterých jsem čerpala.

.....
Aneta Helešicová
22. května 2018

Poděkování

Tímto bych chtěla poděkovat panu Ing. Michalovi Kapinusovi za veškerou pomoc a podporu, kterou mi při psaní bakalářské práce poskytl. Zároveň bych chtěla poděkovat i rodičům, kteří mě vždy podporovali, přátelům, za jejich konstruktivní i nekonstruktivní kritiku mých postupů a za ochotu při testování, a skvělému příteli, který byl ochotný se mnou procházet kódy pokaždé když program přestal fungovat a nenechal mě psaní bakalářky vzdát. Největší poděkování ale patří mé sestře, jejíž nezlomný duch je mi vždy velkou inspirací, a která během doby, kdy tato práce vznikala, vyhrála svůj nejdůležitější boj.

Obsah

Kapitola 1

Úvod

> „Hra je dobrovolná činnost, která je vykonávána uvnitř pevně stanovených časových a prostorových hranic, podle dobrovolně přijatých, ale bezpodmínečně závazných pravidel, která má svůj cíl v sobě samé a je doprovázena pocitem napětí a radosti a vědomím jiného bytí než je všední život.“
>
> Johan Huizinga, *Homo Ludens*, 1938

Obliba deskových her v poslední době stále roste. Věřím, že právě deskové hry pomáhají rozvíjet kreativitu, schopnost řešení konfliktů, uvolňují stres, umožňují rozvíjet taktické schopnosti, postřeh, všímavost, donutí nás komunikovat a mají mnoho dalších skvělých pozitivních vlivů na lidskou osobnost, jelikož si myslím, že pro každičký aspekt lidské osobnosti existuje hra, jež jej pomůže rozvíjet.

Důkazem rostoucí obliby může být i stále narůstající počet deskoherních klubů po celé republice. Některé hry ale mohou být příliš komplikované a zdlouhavé, což odrazuje potenciální hráče. Příkladem takové hry může být *World of Warcraft Desková hra*, které se budu v této práci věnovat.

V této práci se snažím problém vyřešit pomocí rozšířené reality, která hráčům dokáže napovědět a pomáhá hru urychlit a zjednodušit. Naprogramuji tedy hru v jazyce C++ s využitím Robot Operating System, Qt a databázemi v SQLite.

Jelikož se jedná o mé první setkání s jakýmkoliv druhem virtuální reality, na prvních stranách této práce v kapitole druhé „Teoretická část“ popisuji, co jsem musela nastudovat k tomu, abych dokázala téma správně uchopit a řešit. Protože jsem měla hned od začátku určenou platformu, pro kterou má být tato práce napsaná, věnuji se v této kapitole i jí. Platforma ARTable mě svými specifikacemi vázala na určité frameworky a další nástroje a jiné jsem si sama zvolila, proto v teoretické části zmiňuji i je.

Postupně přecházím k návrhu práce (kapitola třetí), který zahrnuje i popsání hry, kterou do rozšířené reality převádím, aby mohl čtenář lépe pochopit jaké problémy a otázky bylo nutné řešit. Po návrhu hovořím o výsledné realizaci kódu aplikace, zmiňuji některé specifické problémy vázané ke stanovené platformě a k rozšířené realitě v kapitole čtvrté.

Na konci jsem krátce zmínila testování aplikace a změny na základě něj provedené, a v závěru shrnuji nejdůležitější poznatky práce a zhodnocuji její výsledek.

Kapitola 2

Teoretická část

V této kapitole jsou rozebrána základní témata práce, která tvoří oblast rozšířené reality, její definici a využití, uživatelská rozhraní se zaměřením na promítaná uživatelská rozhraní, způsoby kterými se toto uživatelské rozhraní zobrazuje a ovládá, část je věnována také platformě ARTable a frameworku Robot Operating System.

2.1 Rozšířená realita

Rozšířená realita (zkratkou AR z anglického *Augmented Reality*) je dle článku Ronalda T. Azumy[4] druh virtuálního prostředí. Virtuální prostředí ovšem při použití nedovoluje uživateli vůbec vidět reálný svět. Oproti tomu v rozšířené realitě uživatel vidí reálný svět s virtuálními objekty umístěnými v něm. Takže rozšířená realita doplňuje realitu místo toho, aby ji kompletně nahrazovala.

Rozšířenou realitu definujeme pomocí tří vlastností:

1. Kombinuje realitu a virtuální svět
2. Je interaktivní v reálném čase
3. Existuje ve 3D

Tyto vlastnosti plně odpovídají tomu, co je základní myšlenkou implementace v mé práci.

Využití

Využití rozšířené reality se stává stále běžnější součástí každodenního života. Rozšířená realita se nejdříve používala pro účely průmyslu, armády a lékařství, stále více se ale dostává do běžnějších a běžnějších situací v našich každodenních životech, kupříkladu v odvětví zábavy, typicky videoher, nebo vzdělávání.

Průměrný člověk se pravděpodobně k rozšířené realitě dostane nejlépe prostřednictvím zábavy, například ve formě různých aplikací na chytré telefony či tablety. Jedním z příkladů může být hra Pokémon GO, která v roce 2016 slavila obrovský úspěch svým přístupem k rozšířené realitě. Jedná se o aplikaci na mobilní telefony, která rozšířenou realitu zábavnou formou dostala k milionům uživatelů, způsobem, který je vidět na obrázku 2.1. Jelikož se jednalo o první takto masovou aplikaci využívající rozšířenou realitu, zdvihla velkou vlnu zájmu.

Obrázek 2.1: Hra Pokémon GO, ve které se animované postavičky pokémonů zobrazují v reálném světě.

Další z oblastí využití je vzdělávání. Sada aplikací pro chytré telefony se postupně rozrůstá a hojně jsou mezi nimi zastoupeny i aplikace vzdělávacího charakteru. Umožňují tak například vizualizovat obrázky z učebnic. Zajímavým využitím je také takzvané VirtualiTee - tričko s potiskem, na který se zamíří kamera telefonu se spuštěnou aplikací, která podle potisku zvládne vizualizovat vnitřní orgány člověka, viz obrázek 2.2.

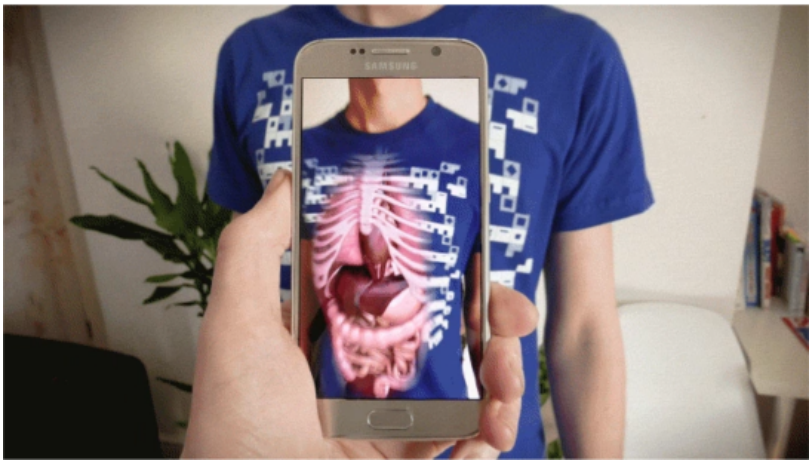
Z těch oblastí, ke kterým se už člověk nedostane, pokud nesouvisí s výkonem jeho zaměstnání, se jedná například o medicínské využití, využití v dílnách a v armádě.

Medicínské využití může úzce souviset s výukou, při které umožňuje například vizualizaci vnitřních orgánů, nebo usnadňovat práci operatérům, kterým dokáže rozšířená realita zvětšovat obraz na který se aktuálně dívají, nebo vizualizovat jakým způsobem nejlépe zavést jehlu [4][7].

Zobrazovací zařízení

Pro zobrazení virtuální reality se používají různá zařízení. Může se jednat například o *Head-mounted display* (HMD) - zařízení, které má uživatel umístěné na hlavě, sledující jeho pohyby, pomocí čehož může měnit obraz promítaný na optický displej zařízení. Typickým zástupcem HMD jsou brýle, například Vuzix AR3000 AugmentedReality SmartGlasses či Google glass (na obrázku 2.3).

[2]Převzato z https://www.kickstarter.com/projects/curiscope/virtualitee



Obrázek 2.2: Tričko a aplikace Virtuali-Tee, která snímá značky, které jsou na tričku natištěné, a na základě nich vizualizuje vnitřní orgány v rozšířené realitě.[2]

Obrázek 2.3: Google Glass Enterprise Edition[3]

Nejdostupnější zařízení, které dokáže zobrazovat virtuální či rozšířenou realitu, je klasický chytrý telefon nebo tablet. Jeho využití můžeme sledovat v mnohých projektech pro virtuální realitu, které se snaží cílit na širokou veřejnost, a není pro ně až tak důležitá kvalita oddělení reálného světa od toho virtuálního, aby po uživateli požadovaly vlastnictví HMD.

Z opačné škály co se dostupnosti týče, bych vyzdvihla studio Théoriz[4], které pro zobrazování virtuální reality využívá celou místnost, ve které jsou umístěny projektory a senzory pohybu, takže se může celý pokoj měnit na základě toho, jak se člověk v něm pohybuje, viz obrázek 2.4.

Prostorová rozšířená realita

Prostorová rozšířená realita, nebo také *Spatial Augmented Reality* (SAR) je zvláštní druh rozšířené reality. V předchozí podsekci o zobrazování virtuální reality jsem zmínila možnost zobrazovat virtuální realitu ne pouze v omezeném prostoru nějakého zařízení, ale v prostoru celé místnosti. To právě je nazýváno prostorovou rozšířenou realitou [6].

[3] Převzato z https://www.svetandroida.cz/bryle-google-glass-se-daji-koupit/
[4] http://www.theoriz.com/
[5] Převzato z http://www.theoriz.com/

Obrázek 2.4: Projekt smíšené reality - studio Théoriz [5]

K jejímu vytvoření se využívá různých druhů prostorových displejů, což můžou být jakékoliv plochy, které slouží k zobrazování SAR, a nejsou závislé na uživateli, nýbrž na okolním prostoru. K tomuto účelu mohou sloužit jak klasické displeje, tak speciální průhledné obrazovky, hologramy, či běžné projektory.

Nejčastěji jsou využívány právě projektory, jelikož jsou snadno přenositelné a nemusí sloužit k pouze jednomu účelu. Moderní „chytré“ projektory už také dokáží pracovat s povrchem, na který promítají, a provést korekci obrazu tak, aby okolní prostředí nikterak nenarušovalo dojem z promítaného obrazu. Využívají k tomu algoritmy pro úpravu barev i tvaru promítaného obrazu tak, aby vypadal co nejpřirozeněji. Podrobněji jsou tyto úpravy popsány v článku „Embedded entertainment with smart projectors“, ze kterého je také převzat obrázek 2.5, který ukazuje rozdíl mezi upraveným a neupraveným obrazem na nedokonalém povrchu [5].

2.2 Uživatelská rozhraní

Uživatelské rozhraní můžeme definovat jako způsob, kterým uživatel komunikuje se systémem. Jedná se o jednu ze součástí Human-Computer interaction (HCI). Zpříjemnění používání aplikace je jedním z nejožehavějších témat návrhů uživatelských rozhraní. Vývoj směrem k co nejintuitivnějšímu a člověku nejbližšímu ovládání je znatelný.

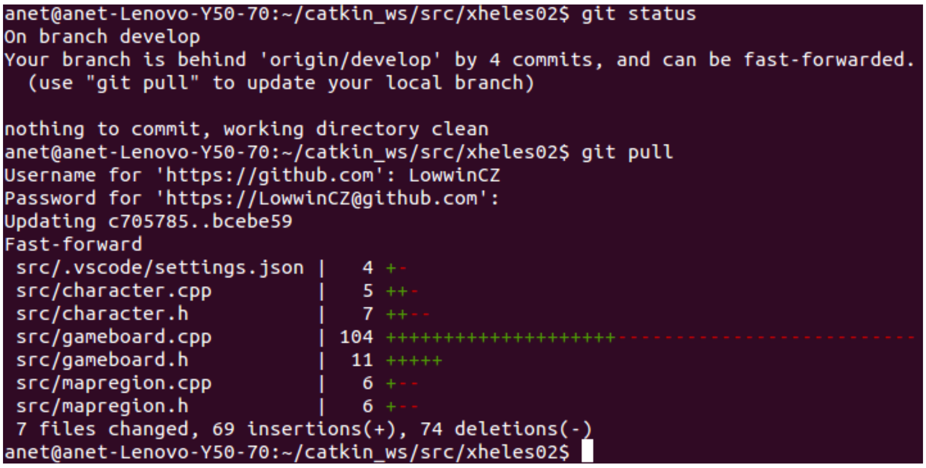
Typy uživatelských rozhraní

Tři základní typy uživatelských rozhraní, které se dnes využívají, jsou definovány v následujících řádcích, seřazené podle data vzniku.

Textové uživatelské rozhraní - CUI (Character User Interface)
Nejstarší používané uživatelské rozhraní je to textové. Veškeré jeho ovládací prvky jsou složeny pouze z textu či znaků. V dnešní době je jeho využití nemyslitelné pro ovládání většiny programů běžnými uživateli. Stále se ovšem s tímto rozhraním můžeme setkat, typické použití je například v příkazové řádce. Běžně používaný příklad může být používání gitu v příkazové řádce, viz obrázek 2.6.

Grafické uživatelské rozhraní - GUI (Graphical User Interface)
Dnes nejběžněji používané rozhraní je jednoznačně grafické uživatelské rozhraní. To využívá různé grafické prvky jako okna, tlačítka a podobně, kombinované s textem. Uživatel

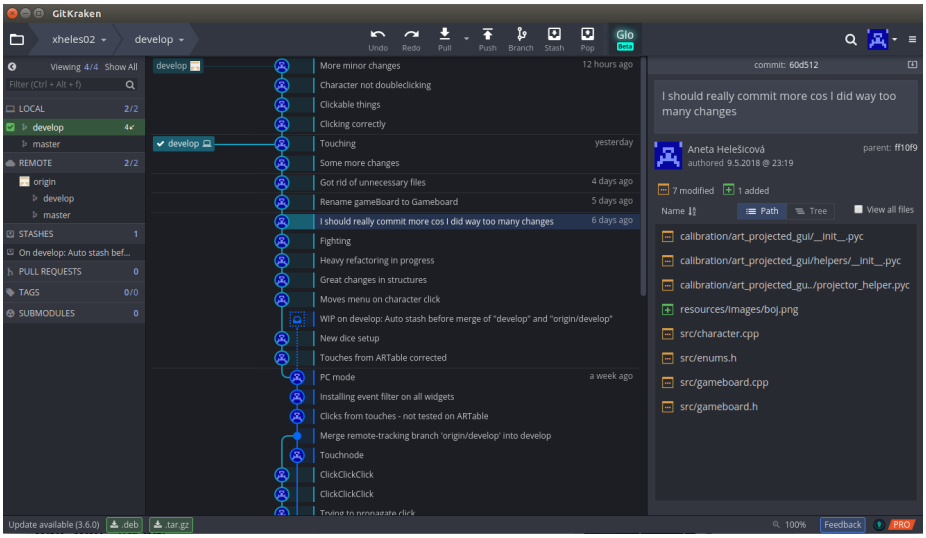
Obrázek 2.5: Projekce chytrého projektoru na nedokonalý povrch. Na obrázku (a) je vyobrazen povrch, na který je ve zbylých obrázcích promítáno. Patrné jsou nestejné barvy a povrchy a to, že se jedná o roh místnosti. Obrázek (b) pak zobrazuje jak vypadá promítaný obraz bez jakýchkoliv korekcí. Na obrázku (c) dojde k úpravě geometrie obrazu tak, aby nebylo patrné promítání na nerovnou plochu a na posledním obrázku (d) je upravena i barevnost promítaného obrazu na základě barvy podkladu.



Obrázek 2.6: Textové uživatelské rozhraní pro službu GIT v terminálu na operačním systému Ubuntu.

rozhraní ovládá typicky pomocí nástrojů jako myš či klávesnice, které snímají jeho pohyby. Pohyby ovšem nejsou přirozené, jen uzpůsobené ovládání počítače. Velkou výhodou je jeho nenáročnost pro uživatele i pro programátory. Pro srovnání s textovým rozhraním

přikládám obrázek 2.7 programu GitKraken, který je grafickým uživatelským rozhraním pro službu GIT a umožňuje tedy stejnou funkcionalitu jako příkazová řádka z obrázku 2.6.



Obrázek 2.7: Grafické uživatelské rozhraní programu GitKraken.

Přirozené uživatelské rozhraní - NUI (Natural User Interface)

Přirozené uživatelské rozhraní mají několik základních znaků. Jejich používání je intuitivní, velmi často bývají flexibilní (takže si je dokáže uživatel přizpůsobit k obrazu svému) a jejich používání je velmi plynulé, uživatel si totiž nemusí ani uvědomovat, že používá uživatelské rozhraní.[13]

NUI umožňuje uživateli ovládat systém pomocí hlasu, gest či doteku. Podle těchto základních způsobů ovládání se tedy dále dělí na rozhraní hlasové (Voice UI) či hmatové (Tangible UI).

Hlasové rozhraní je zjevně ovládáno pomocí hlasu. Tento způsob ovládání se stává čím dál tím běžnější, můžeme se s ním setkat například při používání hlasových asistentů, jejichž zástupcem může být Google Home či Amazon Echo.

Hmatové rozhraní umožňuje uživateli ovládání virtuálního prostředí skrze fyzická zařízení reálného světa, která nejsou přímo určena k ovládání počítače. Zajímavým příkladem takového rozhraní může být „The Reactable“, což je elektro-akustický hudební nástroj, který se ovládá skrze hmatové rozhraní. Toho rozhraní je zde provedeno jako stůl, jehož desku tvoří displej. Na něj se pokládají různé typy bloků, takzvaných „tangibles“, které Reactable snímá a podle jejich pozic ovlivňuje vydávaný zvuk.[8]

Promítané uživatelské rozhraní a jeho ovládání

Promítané uživatelské rozhraní by se dalo na základě předešlého dělení zařadit k hmatovému rozhraní, kdy prostředkem pro ovládání je přímo ruka uživatele.

Je to takové uživatelské rozhraní, které je zobrazováno promítáním. K jeho ovládání mohou sloužit například dotykové plochy, na které je promítáno, či různé typy snímání okolí pomocí senzorů.

2.3 Způsoby zobrazení uživatelského rozhraní

Přímo téma práce specifikuje, že vývoj aplikace bude probíhat na dotykový stůl. Ačkoliv platforma ARTable již existuje, chtěla jsem se zaměřit na to, jakým způsobem by se daly některé součásti dělat jinak.

Jako první jsem se zaměřila na možnosti zobrazování grafického uživatelského rozhraní.

Projektor nad zobrazovací plochou

ARTable využívá pro zobrazování uživatelského rozhraní projektor umístěný nad plochou stolu, což se zdá jako nejjednodušší, nejlevnější a nejpřímočařejší řešení problému zobrazování. Při tomto způsobu zobrazování ale ruka hráče tvoří při ovládání na hrací ploše stíny, což mi nepřijde zcela ideální [10].

Projektor pod zobrazovací plochou

Tvorba těchto stínů by odpadla, promítalo-li by se na stůl zespod. Toto řešení by ovšem vyžadovalo něco jiného než běžný kancelářský stůl a na něm umístěnou neprůsvitnou dotykovou desku. Dá se předpokládat, že by cena této platformy mnohonásobně vzrostla, a kvůli pouhé tvorbě stínů při hraní nemá smysl investovat do řešení, které by tento způsob promítání dokázalo podporovat [10].

Využití monitoru

Kdyby se namísto promítání uživatelského rozhraní využilo dotykového monitoru, získal by ARTable mnohem lepší obraz, čímž je myšleno vyšší DPI, lépe vykreslené barvy a větší ostrost obrazu, dá se ale předpokládat, že monitor, který by byl použit by byl v porovnání se současným řešením buď velmi malý (aktuální rozměry dotykové plochy ARTable jsou 100 cm na 60 cm), nebo velmi drahý.

Porovnáním výhod a nevýhod jednotlivých možností zobrazení jsem došla k závěru, že vzhledem k požadovaným vlastnostem vyvíjeného interaktivního stolu je způsob promítání shora zvolen zcela smysluplně. Pokud by se požadavky na tento stůl změnily, bylo by možná vhodné popřemýšlet o zkvalitnění systému zobrazování.

2.4 Způsoby ovládání uživatelského rozhraní

Další částí systému, jejíž alternativy jsem chtěla zjistit, bylo ovládání zobrazeného uživatelského rozhraní. ARTable k tomu využívá kombinace dvou prvků - dotykové plochy a kamery, která dokáže snímat objekty na stole a osoby u něj.

Dotyková plocha

I v dotykových plochách existují různé typy řešení a na různých fyzikálních principech fungující snímání doteku. Pro ARTable bylo zvoleno řešení pomocí kapacitní dotykové folie, na které je umístěn speciální vodivý plast [2].

Kapacitní dotyková plocha

Fyzikální princip kapacitní dotykové plochy je v podstatě jednoduchý - dotek je zaznamenán, jakmile přijde plocha do styku s vodivým předmětem, jako je například lidský prst.

Díky tomu je plocha velice citlivá, dotek tudíž nemusí být nikterak silný aby byl zaznamenán, což může být jak výhoda, tak nevýhoda. Vysoká citlivost totiž znamená vyšší míru překlepů při rychlém používání (například psaní na klávesnici). Tento typ dotykové plochy také podporuje zaznamenávání více doteků zároveň [9][11].

Odporová dotyková plocha

Společně s kapacitní dotykovou plochou se jedná o špičku v dotykových plochách. Odporová plocha však funguje na kompletně jiném fyzikálním principu, neboť zatímco u kapacitní dotykové plochy je vytvářeno spojení vodivým předmětem, zde dotek můžeme provést čímkoliv, a spojení proběhne mezi dvěma vrstvami plochy - pevnou zadní a flexibilní přední vrstvou. Jelikož musí dojít k dotyku mezi těmito vrstvami, je nutné působit na plochu větší silou [9][11].

Dotyková plocha na principu povrchové akustické vlny

Povrchová akustická vlna je taková akustická vlna, která se šíří podél povrchu pružného materiálu. Této vlny využívá další typ dotykové plochy, kdy jsou na okrajích této plochy umístěny akustické převodníky, které na ploše vytváří pole ultrasonických vln. V momentě, kdy se pak uživatel dotkne plochy, je toto pole narušeno, z čehož se dá zjistit dotyk i jeho pozice [11].

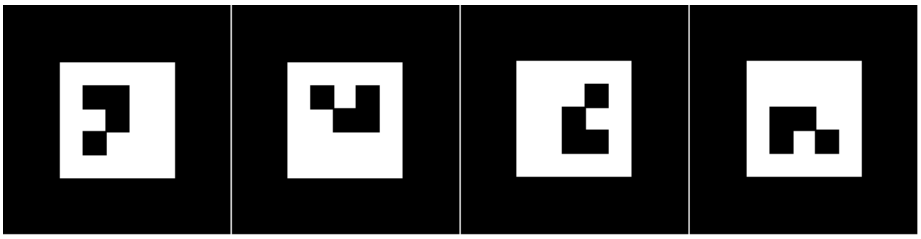
Infračervená dotyková plocha

Tento typ dotykové plochy má po okrajích umístěny vysílače a přijímače infračerveného záření. Princip je jinak obdobný jako u dotykových ploch s povrchovými akustickými vlnami - při dotyku taktéž dojde k narušení mřížky těchto paprsků, z čehož odvodíme polohu dotyku [11].

Snímání gest a objektů pomocí kamery

Snímání objektů pomocí kamery může probíhat ve dvou variantách, první variantou je snímání založené na značkách (marker-based) a druhou snímání bez značek (markerless).

Snímání založené na značkách pracuje s tím, že předem ví, které věci má v obraze hledat. Značky mohou být různé, typicky se ale jedná o černobílé čtverce s různě velkými mřížkami, jejichž pole jsou vyplněna černě či bíle. Na obrázku 2.8 jsou zobrazeny typičtí zástupci těchto značek. Platí přitom, že čím jednodušší a větší značky jsou, tím snadněji a přesněji jsou v obraze nalezeny.



Obrázek 2.8: Běžně používané značky pro *marker-based* rozšířenou realitu.[6]

Oproti tomu snímání bez značek nepotřebuje žádné druhy značek k rozpoznání hledaného objektu. To je ale mnohem náročnější na implementaci, neboť program předem neví, jak přesně bude předmět, který vyhledává v obraze, vypadat.

[6]Převzato z http://tamarisk.io/articles/images/article_data/ar_vr/ar_markers/rotated-ar-markers.png

2.5 ARTable

V této sekci je krátce popsána platforma ARTable, pro kterou celá bakalářská práce vzniká.

ARTable je, jak už název napovídá, "stůl pro rozšířenou realitu". Vzniká pod výzkumnou skupinou Robo@FIT na Fakultě informačních technologií Vysokého učení technického v Brně. Propojuje dotykový stůl, robota, kamery, kinecty a projektor do jednoho systému. Pro svou práci robota nebudu využívat. Zbylé zařízení mi ovšem umožní vytvořit na stole 2D promítané grafické uživatelské rozhraní v rozšířené realitě. Jakým způsobem která zařízení používám je součástí implementačního popisu dále v práci.[7]

Konkrétní verze této platformy, kterou využívám pro svou práci, je ARTable3. Jedná se o asi nejmenší verzi ARTable, která oproti „velké“ verzi hlavně neobsahuje žádného robota. Skládá se tedy z klasického stolu, přenosné dotykové desky, projektoru Acer K132, Kinectu One a stolního počítače, viz obrázek 2.9.

Na počítači ARTable3 je operační systém Ubuntu 16.04, systém pro sestavení zdrojových souborů `catkin` a framework `Robot Operating System` ve verzi Indigo.

Obrázek 2.9: Sestava stolu ARTable3.

2.6 Robot Operating System

Robot Operating System, zkráceně ROS, je dle slov jeho autorů „Flexibilní framework pro psaní softwaru pro roboty.“ Jedná se o kolekci nástrojů, knihoven a konvencí, jejíž cílem je zjednodušit tvorbu komplexního a robustního robotického chování na celé škále různých robotických platforem [1]. Jedná se o open source framework, doporučovaným operačním

[7] Zdrojové kódy a bližší informace na stránce https://github.com/robofit/artable

systémem pro použití tohoto frameworku je Ubuntu, pro které jsou vydávány předkompilované balíčky.

ROS byl vyvinut ve Stanfordské laboratoři umělé inteligence, dnes je hojně využíván programátory po celém světě pro vzdělávací i komerční účely. Díky způsobu, kterým ROS funguje, je snadné přispět novými balíčky, které ROS rozšiřují a vylepšují, proto se ROS drží na špičce frameworků v této oblasti.

Základní principy a myšlenky frameworku ROS

Filozofické principy by se daly shrnout do pěti krátkých bodů: [12]

- *Peer-to-peer* - Systém, který byl vytvořen pomocí tohoto frameworku, se bude skládat z několika různých procesů, které by mohly potencionálně mohly běžet zároveň na několika hostech, propojených peer-to-peer sítí. Centrální data server by totiž mohl působit problémy v heterogenních sítích, se kterými se většinou pro rozsáhlé, ROS využívající systémy, počítá. Toto řešení sice vyžaduje nějakou režiji, o kterou se stará takzvaný ROS Master, nebo jmenný server, ale umožňuje nepoužívat centrální server.

- *Vícejazykový* - Tak, jako snad každý programátor, měli i tvůrci tohoto frameworku určité preference ohledně programovacích jazyků. Proto je ROS navržen tak, aby byl jazykově nezávislý. Ve článku uvádí autoři podporu pro jazyky C++. Python, Octave a LISP, další, novější texty však zmiňují i jazyky Java, Lua a dokonce i podporu pro Matlab.

- *Založený na nástrojích* - Z anglického „tools-based“, tento bod jednoduše mluví o tom, že ROS využívá raději mnoha různých nástrojů, které si každý programátor může do svého systému poskládat sám, než by vytvořil velké a těžkopádné monolitické vývojové prostředí.

- *Malý* - V úzké souvislosti s předchozím bodem je i tento bod. Jelikož je ROS roztříštěn do velkého množství nástrojů, snaží se podporovat, aby tyto nástroje zůstaly na ROSu zcela nezávislé a mohly se tak využít i v jiných systémech pro vývoj robotů. Doporučují tedy oddělování co největšího množství samotných výpočtů separovat do nových knihoven, jež poté dokáží na ROS připojit pomocí nějakého jednoduššího rozhraní.

  ROS samotný využívá již v základu některé open source knihovny, co se konkrétně této práce týče, jde například o knihovnu OpenCV pro zobrazovací algoritmy.

- *Zdarma a open source* - Celý zdrojový kód tohoto frameworku je veřejně dostupný a jeho tvůrci věří, že vývoj ROSu jako open source je jediná správná cesta, pokud chtějí paralelně vyvíjet tento systém na mnoha různý hardwarových i softwarových úrovních.

Důležité pojmy a funkční principy

Celý systém rosu funguje na principu jednotlivých *uzlů*, takzvaných „node“, což je výraz se kterým pracuji i v pozdější části práce. Každý z těchto uzlů vykonává určitou konkrétní, přesně definovanou činnost ve svém vlastním procesu. Má-li o to zájem, může veřejně do systému publikovat *zprávy*.

Zprávy jsou definovány speciálními, velice jednoduchými soubory s příponou msg, které určují co která zpráva obsahuje za informace, a které jsou kompatibilní s jazyky, které ROS

využívá. Podporovány jsou primitivní datové typy, jejich pole a konstanty. Dále může zpráva obsahovat jiný typ zprávy, případně pole zpráv. Zanořování zpráv může být libovolně velké. Pro názornost taková zpráva může vypadat následovně:

```
int32 id
bool touch
geometry_msgs/PointStamped point
```

Tato zpráva obsahuje tři parametry - jedná se o integer id, boolovskou hodnotu touch a zanořenou zprávu geometry_msgs/PointStamped point. Definice této zanořené zprávy udává, že se skládá z dalších dvou zpráv - std_msgs/Header a geometry_msgs/Point, kdy každá z nich nese další tři hodnoty. Ukázkovou zprávou byla zvolena Touch.msg, zpráva z ARTable, která slouží k informaci o aktuálním doteku na dotykové ploše.

Pro výměnu dat existují dva přístupy - synchronní a asynchronní.

Asynchronní zasílání a přijímání dat je jednodušším z těchto dvou způsobů. V ROSu je tento přístup zajištěn publikováním zpráv na určitá *témata*, v anglickém originále „topics“. Uzel, který zprávy vytváří a do tématu zasílá, se nazývá „publisher“. V jiné části systému pak bude existovat uzel nazvaný „subsciber“, který bude na tématu, které jej zajímá, naslouchat, a zpracovávat zprávy, které budou do tohoto tématu publikovány. Publisherů do jednoho tématu i subsciberů na něm naslouchajících může být neomezené množství.

Synchronní výměnu zpráv zajišťují *služby*, neboli „services“, které jsou definovány názvem a párem druhů zpráv, jeden typ pro požadavky a druhý pro odpovědi. Zřejmá je zde podobnost s web servery definovanými pomocí URI. Na rozdíl od schématu publikovaní / naslouchání je tento systém vhodný pro schéma Remote Procedure Call požadavek / odpověď interakce, které jsou často využívány v distribuovaných systémech.

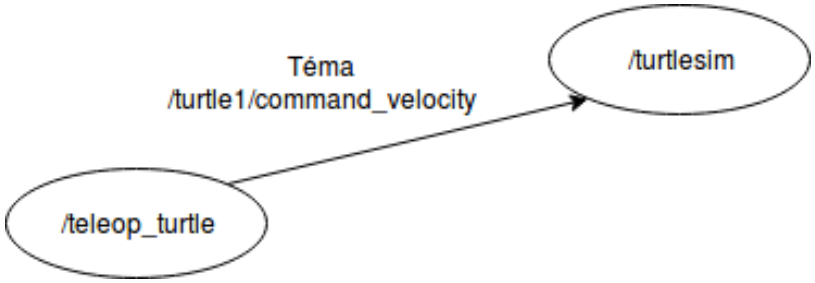
Služby mají, podobně jako zprávy, vlastní typ souboru, ve kterém jsou definovány, tentokrát se jedná o soubor s příponou srv. Koncept těchto souborů je prakticky stejný jako soubory pro zprávy, s tím rozdílem, že definuje dva typy zpráv zároveň.

Uzel může služby využít tak, že pošle požadavek na uzel, kde byla služba definována, a následně čeká až obdrží odpověď.

Aby mohly uzly mezi sebou komunikovat, je samozřejmě nutné, aby se dokázaly najít v rámci systému. Proto existuje jakýsi hlavní prvek, takzvaný *Master*, který se svým způsobem chová jako jmenný server (server, který uchovává informace o vztahu mezi doménovým jménem a IP adresou a tím umožňuje vzájemné převody mezi těmito dvěma informacemi). Uchovává také informace o existujících tématech a službách. Proto se uzel, který se pokouší spojit s jiným uzlem, nejprve obrátí na mastera, který mu poskytne informace o tomto uzlu. Následující komunikace mezi uzly už funguje bez prostředníka.

Pro praktickou ilustraci fungování uzlů slouží následující obrázek 2.10 a popis. Základní tutoriály ROSu pracují s ovládáním želvy v uzlu „turtlesim“. Tento uzel zobrazuje velmi jednoduché grafické rozhraní, které je vlastně jen želva v poli. Zároveň publikuje zprávy na některá témata a spravuje některé služby, to, co je ovšem důležité pro tento příklad, je, že naslouchá na téma „turtleX/command_velocity“. Tato želva se dá ovládat pomocí šipek, jejichž stisk zaznamenává a zpracovává uzel „teleop_turtle“. Po zpracování takového stisku publikuje uzel zprávu obsahující informace o pohybu (lineární pohyb a úhel). Turtlesim si poté tuto zprávu přečte, zpracuje a na jejím základě změní vykreslení želvy v okně.[8]

[8] Tutoriál se pro novější verze mírně liší.



Obrázek 2.10: Zobrazení uzlů pomocí turtlesim tutoriálu

2.7 catkin

S frameworkem ROS úzce souvisí jeho oficiální systém pro sestavování programů (anglicky *build system*) `catkin`, který je následovníkem původního systému `rosbuild`.

Umožňuje generování cílů ze zdrojových kódů poskytnutých v jednotlivých ROS balíčcích. Zároveň spolupracuje s jinými, podobnými systémy. Ve své práci využívám kompatibility catkin s CMake.

2.8 Qt

Qt je framework pro tvorbu grafických uživatelských rozhraní pro programy psané v jazyce C++ (případně pro jazyk Python existuje varianta PyQt). Mimo framework samotný nabízí i velmi příjemné vývojové prostředí, kde umožňuje i tvorbu vzhledu aplikace intuitivně pomocí přidávání prvků do aplikace jednoduše graficky jejich přetáhnutím z nástrojové lišty na požadované místo.

Qt podporuje mnoho platforem, jak desktopové Windows, Linux i macOs, tak jejich mobilní verze (iOS, Windows Phone), tak i Android a jiné platformy.

Nabízí mnoho různých tříd a prvků umožňující tvorbu od jednoduchých 2D aplikací až po 3D aplikace, aplikace pracující s videi, animacemi a podobně.

Nejaktuálnější verzí je Qt 5.9, já ve své práci využívám Qt ve verzi 5.1 pro zajištění kompatibility.

Kapitola 3

Návrh

Ještě před začátkem programování samotné hry bylo potřeba ujasnit si, co přesně budu dělat, čeho tím chci dosáhnout a jak toho chci dosáhnout. S tím souvisí především pravidla originální hry, která se chytám implementovat, a způsob jak je chci implementovat.

3.1 Originální hra

Hra *World of Warcraft Desková hra* je jedna z nejrozsáhlejších her, se kterými jsem kdy přišla do styku. Aby bylo možné pochopit řešení práce, pokusím se zde ve zkratce nastínit pravidla a způsob hraní originální hry.[1]

Základní herní principy

Hra je koncipována pro 2-6 hráčů. Ti jsou rozdělení do dvou týmů, také nazývaných frakcemi, (Aliance a Horda) a každý z nich ovládá 1-3 postavy, do maximálního počtu šesti postav celkově ve hře. V rámci týmu musí hráči a postavy kooperovat, zároveň se snaží porazit druhý tým. Celá hra je o plnění úkolů na mapě, sbírání zkušenostních bodů, vylepšování postav, soubojích a závěrečném poražení takzvaného overlorda.

Hra funguje v kolech, každé kolo náleží jedné frakci a každá postava této frakce v něm má dvě volné akce. V každém kole se také může podle symbolů na počítadle kol stát nějaké akce - přidávání předmětů do obchodu, nebo losování speciální karty kola.

Herní plán

Jedním z nejdůležitějších prvků hry je herní plán. Jeho většinu tvoří mapa, po které se postavy pohybují, protože plnění úkolů je vždy vázané k určitému kraji. Na herním plánu jsou také umístěné počítadla - počítadlo zkušenostních bodů a počítadlo kol, a oblasti pro souboje. Jednotlivé kraje se mohou lišit svými vlastnostmi - mohou obsahovat města, letecké linky a dá se z nich přemístit jen do několika okolních krajů.

Na mapě se všichni hráči a nestvůry zobrazují figurkami, některé další informace ohledně krajů pomocí žetonů.

[1] Celé znění originálních pravidel v angličtině lze nalézt na stránce https://images-cdn.fantasyflightgames.com/ffg_content/WoWBG/wowrules.pdf.

Karta hráče

Každé postavě náleží karta, ze které je hráč schopný vyčíst veškeré potřebné informace o postavě. Kromě údajů jako zdraví postavy a množství energie, se na kartu ukládají veškeré schopnosti a talenty postavy a také předměty, které postava aktuálně používá.

Obrázek 3.1: Karta hráče

Úkoly

Postavy získávají zkušenostní body plněním úkolů. Každý tým může mít v jeden moment rozehrané jen tři úkoly jakékoliv složitosti si zvolí (šedá - zelená - žlutá - červená), kromě šedé, se kterou oba týmy začínají a která se nedá dobírat. Po splnění náleží postavě/postavám, které se splnění zúčastnily, odměna ve formě zkušenostních bodů, zlaťáků a občas předmětů. Jeden úkol je vždy na jedné kartě, na které jsou uvedeny veškeré informace - které nestvůry je potřeba porazit ve kterém kraji, jaká je odměna, pro jakou úroveň postavy je úkol určen a případně jaké další nestvůry se na mapě objevily.

Souboje

Poslední součástí, o které bych ráda něco řekla, je soubojový systém. Funguje jak pro boje s nestvůrami, tak s hráči druhé frakce, typicky se ale jedná o první možnost. Každá nestvůra má tři hodnoty, které se dají najít v kartě nestvůr. Tyto hodnoty jsou *h*rozba, útok a *z*draví. Pro poražení je nutné uštědřit tolik zásahů, jakou má nestvůra hodnotu zdraví, či více. Hráči si pro souboj házejí kostkami. Kostky jsou osmistěnné třech typů (červené pro útoky na blízko, modré pro útoky na dálku a zelené pro obranu) a jejich počet se odvíjí od vlastností postavy, jejích schopností, kouzel, talentů a používaných předmětů. Na začátku každého kola boje si hráč vybere které předměty, kouzla a talenty bude používat a podle toho si zvolí kostky, kterými hází. Pro platný hod útoku je potřebné, aby na kostce padla

Obrázek 3.2: Karta úkolu

hodnota stejná nebo vyšší než hrozba nestvůry. Pokud se postavám nepodaří nestvůru zabít v jednom kole, nestvůra jim uštědří zranění v hodnotě jejího útoku. Toto zranění si mohou postavy účastnící se boje libovolně přerozdělit. Poté následuje další kolo boje, dokud jedna ze stran nezemře.

Souboje se může účastnit jak více postav stejné frakce, tak více nestvůr stejného druhu a typu (typy nestvůr se dělí na neutrální - modré, a úkolové - zelené a červené). S neutrálními nestvůrami musí postavy bojovat jakmile vstoupí do kraje, ve kterém se neutrální nestvůry nacházejí. Boj s úkolovými nestvůrami si hráči musí zvolit a nemohou bojovat s nestvůrami druhé frakce. Za boj s neutrálními nestvůrami hráčům nepřísluší žádná odměna.

3.2 Převod do rozšířené reality

Pro převod hry do rozšířené reality bylo využito několika prostředků. Základním bodem bylo ale rozdělit hru na části, které zůstanou fyzické a ty, které se budou promítat.

Promítané části

Vzhledem k omezené dotykové ploše jsem přistoupila k tomu, že promítat budu primárně hrací plán. S tím souvisí promítání veškerých součástí, které se na něm pohybují. Místo figurek tedy budou na plánu portréty jednotlivých postav a nestvůr, a také veškeré žetony, které by na plán byly jinak umisťovány. Portréty budou reagovat na dotek zobrazením informací o zvolené jednotce.

Další promítanou částí jsou karty úkolů a overlord. Promítání úkolů zjednoduší práci při přidávání nestvůr na mapu, umožní počítat zkušenosti postav a i hráčům usnadní orientaci v plněných úkolech. Overlorda jsem se rozhodla implementovat z podobně praktických důvodů, navíc se jedná o speciální typ nestvůry, která má zvláštní chování, tudíž bude implementace jednodušší. Hráčům se tím zjednoduší hra o další část, a tou je starost o pohyb overlorda.

Následující implementovanou částí jsou karty speciálních událostí. Vzhledem k tomu, že často přidávají žetony na mapu a úzce souvisí s počítadlem kol, je i pro hráče lepší, když se o tyto karty bude starat herní logika samotná.

Poslední z ne-fyzických částí hry budou kostky, které bude počítat hra samotná a hráči bude umožněno je hodit pomocí doteku.

Fyzické části a interakce s nimi

Hlavním prvkem, který ponechávám fyzický, budou karty hráčů a vše, co se na ně umisťuje. Znesnadní mi to sice čtení vlastností jednotlivých postav, ale pro uživatele bude mnohem příjemnější mít veškeré tyto prvky před sebou, než kdyby musel sdílet omezený prostor dotykové plochy.

Hlavním problémem, který řeším, je způsob rozpoznávání obrazu. Karty hráčů jako takové budou mít markery, díky kterým bude snadno rozpoznatelné o jakou postavu se jedná, tudíž bude hráčům stačit pouze ke stolu přijít s kartami zvolených postav, a hra je již sama přidá na hrací plán.

Prvotní koncept rozpoznávání kartiček schopností, talentů a předmětů, byl srovnávání obrazových dat s databází karet. Bohužel se to ukázalo jako neefektivní, tudíž hráči budou sami zodpovědní za správné počítání kostek, které v daném hodu použijí, a za využití všech schopností, které jim karty nabízejí.

Oblasti, která herní logika kontrolovat vůbec nebude je počítání zdraví, energie, zlaťáků a nákup v obchodě, získávání talentů a kouzel a již výše zmíněné počítání kostek.

Kapitola 4

Implementace

Celá práce je implementována pro platformu ARTable, za využití jazyků C++ a Python, s databázovým systémem SQLite, sestavováno systémem catkin využívající CMake a ve frameworku Robot Operating System.

4.1 Konfigurace systému a kompatibilita s ARTable

Konfigurace systému pro spuštění programů pro ARTable je poměrně komplikovaná činnost, kterou podrobně popisuji v příloze B. Ve zkratce je ale důležité mít na počítači nainstalovaný operační systém Ubuntu 13.10 nebo 14.04, Python (verze 2.7.6), ROS Indigo, rosdep, libfreenect2 a ial_kinect2.

Je také nutné vytvořit catkin prostředí a naklonovat do něj ARTable repozitář.

Aby tato práce nebyla jen o naprogramování klasické počítačové hry, je celá hra přizpůsobena k ovládání na platformě ARTable, která je popsána již dříve v teoretické kapitole. Toto ovládání si vynutilo řadu přizpůsobení, ať už mluvíme o zobrazování, nebo o ovládání.

Po spuštění ARTable

Jelikož je ARTable poměrně komplikovaný systém, rada bych hned ze začátku zmínila jakými kroky se člověk může dostat ke zprovoznění stroje pro spuštění kteréhokoliv balíčku s aplikací na ARTable cílenou.

Celý ARTable běží na systému ROS, ve verzi Indigo. Proto je po spuštění samotných hardwarových součástí (konkrétně počítače a projektoru) nutno přistoupit ke spuštění ROSu.

Hlavní, na ARTable nezávislá součást tohoto systému, se spustí příkazem `roscore`. Ten spustí ROS Mastera, který zajišťuje správu všech uzlů, které se v průběhu používání vytvoří. Bez tohoto prvku by žádná ROS aplikace nešla vůbec spustit. Dále spustí server pro správu parametrů, kam se dají ukládat nejrůznější parametry ohledně konkrétního systému, které je potřeba ve fázi běhu programu využívat. Může se jednat o parametry od čísel portů, na kterých běží služby, které chceme využít, až po třeba rozlišení. Poslední částí, kterou roscore zajistí, je logování. Roscore musí běžet v samostatném terminálu po celou dobu práce s ROSem.

Dalším prvkem, tentokrát už specifickým pro ARTable, je spuštění celého ARTable systému. K tomu slouží balíček `art_bringup`. Konkrétně příkazem `roslaunch art_bringup bringup.launch` se spustí uzly, které zajišťují databázi, běh projektoru (konkrétně uzel,

který zajišťuje promítání zaslaného obrazu), ovladače kinectu a dotykové plochy a kalibrační uzel.

Kalibrační uzel sám zajišťuje kalibraci souřadného systému stolu, která funguje na principu snímání AR markerů - přesně definovaných značek, které se rozmístí na předem definovaná místa. V tomto případě se jedná o sadu čtyř markerů, kdy každý z nich je předem přiřazen jednomu rohu dotykové vrstvy, kam se umístí před spuštěním kalibrace, tudíž i před spuštěním celého ARTable systému. Kalibrátor poté tyto markery snímá a dle nich si nastaví souřadný systém stolu. Po dokončení kalibrace je samozřejmě možné a nutné tyto markery umístit pryč z desky stolu. Tato kolekce uzlů musí ovšem také zůstat spuštěná po celou dobu ostatní práce.

V tomto momentu už je systém ARTable připraven pro spuštění aplikace jako takové. V aplikaci samotné ale musí docházet k procesům popsaných v dalších bodech.

Promítání

Součástí ARTable je projektor, pomocí kterého dochází k zobrazování promítaného uživatelského rozhraní na dotykovou plochu.

Prvním krokem k funkčnosti promítání na ARTable je kalibrace projektoru. Tu jsem, ačkoliv jinak je práce psaná v jazyce C++, pro jednoduchost zařídila skriptem v jazyce Python, který je jinak primárním jazykem při psaní funkcionality pro ARTable. Kalibrace projektoru probíhá tak, že se promítne na plochu pod ním šachovnice, ze které se následně spočítá kalibrační matice tak, aby obraz odpovídal dříve vytvořenému souřadnicovému systému stolu.

Způsob, jakým dojde k propojení projektoru a aplikace běžící na počítači u ARTable se pokusím nastínit v následujících řádcích. Již v předchozím bodě jsem zmínila kdy dojde k zapnutí uzlu, který se stará o promítání obrazu, který je na projektor zaslán. Aplikace samotná tedy musí umět tyto obrazy v pravidelných, dostatečně krátkých intervalech, zasílat.

Aby bylo toto umožněno, je zasílání obrazu vyhrazeno zvláštní vlákno programu, implementované jako třída, která dědí z třídy QThread. Toto vlákno se při inicializaci postará o navázání TCP spojení se serverem, který naslouchá zasílaným obrazovým datům, na konkrétním portu.

Aby nedocházelo k dlouhým prodlevám mezi akcí uživatele a reakcí aplikace, ale nedošlo ani k přílišnému zahlcení vlivem velkého množství zasílaných dat, jsou obrazová data zasílána každou sekundu.

Po spuštění mého programu si tedy může uživatel všimnout v terminálu určitých informačních výpisů. Jedním z nich je i hláška „Waiting for projector node“. Pokud se pokusí o spuštění na počítači, ke kterému není žádný projektor prostřednictvím ROSu připojen, bude program pouze čekat na připojení uzlu s projektorem a nikdy nedojde ke spuštění hry samotné.

Reakce na dotykovou plochu

Další z nedílných součástí ARTable, kterou pro svou práci využívám, je dotyková plocha. Její použití je zjevné, reaguje na doteky uživatele a zasílá o nich informace do systému ROS.

Při spuštění bringupu došlo i ke spuštění ovladače dotykové vrstvy, která teď umí tyto informace zaznamenávat. Tyto informace o dotecích jsou ve skutečnosti implementovaný jako ROS zprávy. Nad dotykovou vrstvou poté běží uzel „publisher“, který se stará o zasílání těchto zpráv na správný interface.

Zprávy z dotykové vrstvy jsou pro ARTable definovány jako struktura obsahující informaci o ID prstu, což je podstatné zejména pro ovládání několika prsty, boolovské hodnotě o tom, zda daný dotek stále ještě probíhá, a souřadnicích, na kterých byl dotek zaznamenán.

Pro korektní zprovoznění dotykové plochy ve vztahu k aplikaci je opět nutné provést kalibraci. Kalibrace dotykové vrstvy využívá faktu, že již byl zkalibrován projektor, tudíž si nechá na dotykovou plochu zobrazit body, kterých se uživatel následně dotýká, z čehož dokážeme opět zajistit správnou kalibraci tohoto prvku. Kalibrační skript je společný pro dotykovou plochu i projektor, tudíž je kalibrace projektoru psána také v jazyce Python. Ke kalibraci byl využit již existující skript pro kalibraci ARTable, který byl ale mírně upraven.

V aplikaci samotné je tedy potřeba naslouchat, zda jsou zprávy o doteku zasílány. K tomu slouží uzel, takzvaný „subscriber“ či „listener“, který naslouchá na interface, o kterém ví, že jsou na něj zasílány zprávy, o které má zájem. Předem je známo, že tyto zprávy budou publikovány na interface `/art/interface/touchtable/touch`.

Jelikož musí naslouchat v pravidelných intervalech, a čím kratších, tím lepších, je pro tento uzel vyhrazeno opět zvláštní vlákno, opět dědic QThread. V inicializaci tohoto vlákna se nastaví naslouchání na správný interface, a poté se už jen zvoláním `ros::spin` spustí potenciálně nekonečná smyčka, ve které vlákno naslouchá na zadaný interface.

Jakmile zjistí, že na interface přišla nová zpráva, předá tuto zprávu funkci, jež se postará o její zpracování. Konkrétně můj projekt nezajímají multi-dotyková gesta, ani táhnutí po ploše, reaguje pouze na kliknutí, proto jen naslouchá na zprávy a zapamatovává si souřadnice, na kterých byla zpráva přijata.

Jelikož jsou hodnoty pozic na osách x a y získané ze zpráv vždy zadané v metrech, je nutné je ještě přepočítat. Informaci o tom, jaký přepočet má být proveden, udržuje server parametrů v parametru `/art/interface/projected_gui/rpm`, neboli „Resolution per meter“, který si tedy vlákno musí pomocí příkazu `ros::param::get` získat.

V momentě, kdy pak obdrží zprávu o ukončení doteku, zašle do vlákna, které se stará o běh grafické aplikace, signál, aby na dané místo emulovalo kliknutí myší.

4.2 Statická data, databáze

Jelikož implementuji velmi rozsáhlou hru, která obsahuje velké množství statických dat, jediným rozumným řešením pro uložení těchto dat mi přišla databáze.

K její implementaci využívám SQLite3, která je opravdu malá a výkonná a za jednu z jejich největších výhod považuji, že se nemusí nijak instalovat. Navíc existují různé nástroje, které umožňují zobrazování dat a manipulaci s nimi v přehledném grafickém rozhraní, na Ubuntu jsem k tomuto účelu používala aplikaci Sqliteman[1], na Windows 10 aplikaci DB Browser for SQLite[2].

V aplikaci se ukládají následující typy dat:

Regiony

Mapa, po které se hráči pohybují, je složená z jednotlivých regionů. V tabulce *region* jsou uloženy hodnoty každého kraje, který se na mapě vyskytuje. Těmito hodnotami jsou ID, které tvoří dvou- až čtyřpísmenná zkratka názvu regionu, jméno regionu, jeho „stát“, čímž je myšleno členění regionů podle barev, které je na mapě naznačeno barevnými okraji

[1] http://sqliteman.yarpen.cz/
[2] http://sqlitebrowser.org/

regionů, a pozice X a Y. Tyto pozice slouží k umístění prvního prvku, který se na mapě objeví v daném regionu.

S regiony také souvisí tabulka *neighbours*, která má pouze dva sloupce, do kterých se ukládají ID krajů. Každý řádek pak znamená, že mezi těmito dvěma kraji je možno se přesunout jediným pohybem (během jedné akce cestování postavy je možno využít dvou těchto pohybů). Tato informace slouží k vytvoření grafu, jehož procházením se při běhu aplikace zjišťuje dostupnost jednoho regionu z druhého v rámci jediné akce.

Hrdinové

Hrdinů, neboli postav, je sice ve hře vždy nejvýše šest, hra jich ovšem nabízí 16 různých. V databázi uchovávám o každém hrdinovi jeho jméno, frakci a povolání. V tabulce je také sloupec pro ukládání speciálních schopností každého hrdiny, nakonec ale tato data nevyužívám.

Úkoly

Každá frakce má svou sadu úkolů ve čtyřech různých barvách, pro každou frakci existuje 40 originálních úkolů. Tato tabulka patří k nejrozsáhlejším, neboť se ke každému úkolu musí uchovávat údaje o jeho názvu, barvě, pro kterou frakci je určen, pro jakou úroveň hrdinů a jaké jsou za něj odměny.

K tomu slouží sloupce *Gold* pro počet zlaťáků, *XP* pro zkušenostní body, a sloupce *Triangle1, Triangle2, Square1, Square2* a *Round*, které udávají z kolika předmětů jakého typu si hráč může jeden zvolit.

Například, má-li úkol „Temná spojenectví“ hodnotu 2 ve sloupci Triangle1, hodnotu 1 ve sloupci Triangle2 a zbylé tyto sloupce s hodnotou 0, znamená to, že si hráč lízne nejprve dva předměty se symbolem trojúhelníku z hromádky předmětů, z nich si jeden zvolí a ponechá, druhý zahodí. Poté si lízne znovu ze stejného balíčku, tentokrát už jen jednu kartu, a tu si ponechá. Obdobně fungují sloupce Square pro předměty označené čtvercem a Round pro předměty označené kruhem.

Posledním sloupcem této tabulky je *SpecialItem*, který může a nemusí být vyplněn názvem speciálních předmětů, které hráč za splnění obdrží

Nestvůry

Poslední a nejrozsáhlejší sadou statických dat, která databáze udržuje, jsou data o nestvůrách (a speciálním druhu nestvůry overlorda). Tato data jsou uchována v několika různých tabulkách.

První tabulkou je tabulka *beast_type*, která uchovává hodnoty jednotlivých druhů nestvůr. Jejími sloupci mimo ID, které se skládá ze zkratky typu nestvůry a její barvy, jsou pak právě typ nestvůry, její barva a hodnoty hrozby, útoku a zdraví, které se pro každou kombinaci barva - typ nestvůry liší.

Další tabulkou je *beast_group*. Se sloupci Id, Region, Quest a SpawnedBy udržuje informace o všech skupinách nestvůr, které jsou vytvořeny všemi úkoly. Ve sloupci region určuje pomocí cizího klíče na ID regionu kam se bude skupina nestvůr zobrazovat. Sloupec SpawnedBy určuje který úkol vyvolá vytvoření této skupiny při běhu hry. Sloupec Quest by se mohl s předchozím mírně plést, jeho význam je ale který úkol je splněn při zabití této skupiny nestvůr.

Beast group tabulka ale neudržuje informace o přesných nestvůrách, které se v těchto skupinách nachází. K tomu slouží tabulka *beast*. Ta obsahuje sloupce Id, BeastType, který

určuje cizím klíčem o jaký druh nestvůry z tabulky *beast_type* se jedná, sloupec BeastGroup, který tuto konkrétní nestvůru přiřazuje do skupiny nestvůr a sloupec NumberOf, který udává počet nestvůr tohoto druhu ve skupině.

K tomuto mírně rozsáhlejšímu řešení jsem se uchýlila, neboť jeden úkol může vytvořit 1-2 druhy nestvůr, které mohou, ale nemusí, patřit do jedné skupiny nestvůr.

Vzácné nestvůry, které se ve hře zjeví pouze z akčních karet, označujeme jako *bosse*. Tyto nestvůry kromě svých hodnot útoku, zdraví a hrozby obsahují i informace o odměnách, které získá postava, která bosse porazí. Ty jsou podobné odměnám u úkolů, jsou ale různé pro silnější a slabší frakci.

Speciálním typem nestvůry je také *overlord*, jehož zabití je jeden ze způsobů jak ukončit hru. Jelikož je overlord definován lehce odlišně, než jiné nestvůry, je pro něj vytvořena zvláštní tabulka. Overlordů jsou ve hře tři typy (rozlišené ve sloupci Name), jejich hodnoty se ale mění i v závislosti na počtu postav ve hře, tudíž vždy existuje kombinace typ overlorda - 4/6 hráčů, dle kterých jsou určeny hodnoty hrozby, útoku a zdraví, které jsou stejné jako u nestvůr. Overlord má zároveň i speciální abilitu, jež je popsána ve sloupci Pravidla Boje, ten má ale spíše jen informační hodnotu, aplikace samotná jej totiž nevyužívá.

4.3 Herní logika

Celé pozadí hry je implementováno výhradně v jazyce C++. Herní logika obstarává veškeré promítané komponenty hry, jejich zobrazování ve hře, přidávání a odebírání. Pro tvorbu grafického rozhraní jsem zvolila nástroj Qt, o kterém mi bylo známo, že dobře spolupracuje s mnou využívaným frameworkem. Vzhledem k rozsáhlosti hry byla tvorba herní logiky jednou z nejkritičtějších částí, neboť prakticky každá komponenta má mnohá využití a schopnosti, které ve většině ovlivňují i ostatní části hry.

Začátek hry a základní prvky

Při úplném startu hry se herní logika postará o zobrazení mapy, vygenerování tří náhodných startovních (šedých) úkolů pro každou frakci, náhodné zvolení jednoho ze tří možných overlordů, zvolení správných hrdinů a přiřazení všech vygenerovaných součástí hry do správných regionů na mapě. Největší část této práce je implementována v hlavní třídě `Gameboard`, která je středobodem grafického vlákna.

Úkoly a nestvůry

Generování úkolů probíhá náhodně, na začátku dostane každá frakce tři zcela náhodné šedé úkoly, poté už smí vybírat alespoň jejich barvu, konkrétní úkol je ale opět vybrán náhodně. Ve hře se nemůže opakovat jeden úkol víckrát.

Po vygenerování úkolu dochází k generování všech skupin nestvůr, které obsahuje původní karta, a k jejich zobrazení na mapě.

Úkoly jsou definovány jako instance třídy `Quest`.

Zároveň každá instance úkolu vyvolává generování nestvůr, neboli v reálném kódu tvorbu instancí třídy `Beast`, podle toho, které skupiny nestvůr má úkol vyvolat.

Hrdinové a jejich akce, soubojový systém

Hrdinové se na mapě zobrazují jako ikony s jejich jménem. Po doteku na místo ikony je zobrazeno menu akcí, které může hrdina provést. Těmito akcemi jsou: Pohyb, Boj, Odpočinek, Učení a Akce Město.

Zvolení akce pohyb umožní hrdinovi posun až o dva sousední regiony, případně při regionu s létací linkou na jakýkoliv jiný region se spojením s touto linkou.

Akce odpočinku, učení či města nejsou v kompetenci herní logiky, proto je pouze zobrazena informace o tom, co mohou postavy při těchto akcích provést. Jedná se výhradně o manipulaci s kartami a žetony na kartě postavy, kterou hra nekontroluje. Z důvodu, aby si hráči nemuseli při každé této akci nechat zobrazovat informace, nekontroluje hra počet akcí každého hrdiny, místo toho se spoléhá na tlačítko ukončení kola, které předá ovládání druhé frakci (v době kola jedné frakce není možné ovládat postavy jiné frakce).

Poslední zbývající akcí je akce boj, detailněji je popsána níže.

Hrdinové jsou implementováni ve třídě `Character`, která dědí z třídy QLabel. Na rozdíl od této třídy ale implementuje reakci na klikání myší.

Jelikož se akce hrdinů, které hra implementuje, netýkají pouze hrdinů samotných, ale primárně jejich interakce s herním světem, využívá tato třída zasílání signálů do třídy Gameboard, která pak tyto akce provádí, nebo vynutí jejich provedení v jiných třídách. Pro signály využívám makra `Q_SIGNAL`, jehož použití je velmi snadné. V hlavičkovém souboru stačí definovat `Q_SIGNAL` s deklarací, jakou by měla kterákoliv jiná funkce. Ve třídě, která tento signál plánuje zachytávat, se vytvoří `Q_SLOT` metoda, která už má jak deklaraci v hlavičkovém, tak definici ve zdrojovém .cpp souboru. V některé jiné funkci se poté zavolá funkce `connect`, které se jako parametr předá konkrétní instance třídy, ze které budou signály vysílány, deklarace signálu, instance třídy, jež bude signály přijímat, a `Q_SLOT` metoda, která signál zpracuje.

Poté už stačí pouze v moment, kdy si třída přeje signál vyslat, zavolat makro `Q_EMIT` a jako parametr signál, který má být vyslán, příjemce už si signál odchytí a zpracuje sám. Tímto způsobem jsem se vyhnula nepříjemným sdíleným proměnným.

Akce hrdinů fungují v kódu následovně: instance Character zachytí signál `clicked`, který znamená, že na pozici widgetu byl vyslán signál o kliknutí myší. Emituje tedy signál `characterClicked` a jako parametr předá sebe samu. Gameboard naslouchá těmto signálům ze všech instancí hrdinů, tudíž zaznamená signál a pošle jej ke zpracování do slotu `displayMoves`. Tato metoda zobrazí menu s tlačítky pro výběr akce, kterou si hráč přeje, aby hrdina, jehož jméno se v menu aktuálně zobrazuje, vykonal. Další postup poté záleží na zvoleném tlačítku.

Stiskne-li hráč jedno z tlačítek pro akce, které hra neimplementuje, je hráči zobrazena nápověda pro to, co smí v dané akci vykonat.

Stiskne-li tlačítko akce pohyb, skryje Gameboard veškeré prvky mapy (hrdiny, nestvůry aj.) a zobrazí pouze zkratky či názvy regionů. Hráč následně stiskne jeden z těchto názvů, čímž v instanci třídy `MapRegion` proběhne vyhodnocení, zda je zvolený kraj z kraje, ve kterém se poslední zvolená postava aktuálně nachází, dosažitelný (k tomu se používá průchod jednoduchým grafem, ve kterém uzly tvoří jednotlivé regiony a hranami jsou propojeny ty uzly, jejichž regiony se nachází přímo vedle sebe, nebo jsou spojeny letištní přepravou) a pokud ano, je zavolána třídní metoda hrdiny, která jej přemístí do nového regionu, vyvolá klasické zobrazení prvků mapy a překreslí regiony, ve kterých došlo ke změně v počtu prvků v nich umístěných.

Speciálním případem přesunu je přesun do regionu, ve kterém se vyskytují neutrální nestvůry. V tomto případě je ihned po ukončení pohybu vyvolán souboj se těmito nestvůrami.

Soubojový systém

Poslední možnou akcí je vyvolání souboje. Tato akce nejprve zjistí, zda se v kraji, ve kterém se postava aktuálně nachází, vyskytují nějaké nestvůry. Pokud ne, jedná se o neplatný tah a výběr akce skončí. Pokud ano, je dále zkontrolováno, zda se v kraji nacházejí neutrální nestvůry. Je-li tomu tak, je vyvolán souboj s těmito nestvůrami, hráč nesmí bojovat s jinými v tomto regionu, dokud nejsou neutrální nestvůry poraženy. Pokud ani neutrální nestvůry v regionu nejsou, spustí se buď souboj se skupinou nestvůr, která se v kraji nachází, pokud je v něm pouze jediná, případně dostane hráč na výběr z většího počtu skupin.

Po spuštění souboje je vprostřed hracího plánu vykresleno pole pro souboj. Zde hráč může házet kostkami v počtech, které si sám nastaví, nelze ale překročit maximum sedmi kostek jedné barvy. Pro zvolení počtu kostek je využívání prvku `QSpinBox`. Vzhledem k existenci karet, které umožňují manipulaci s kostkami, případně jejich opětovný hod, jsou hráči umožněny i tyto kroky. Jakmile je s kostkami hotov a zapíše do pole oslabení počet žetonů oslabení, potvrdí hod tlačítkem.

Hra následně vyhodnotí úspěšnost souboje. Pokud počet platných modrých kostek převyšuje, nebo je stejný jako hodnota zdraví všech nestvůr ve skupině, je souboj okamžitě vyhrán.

Výhra souboje zapříčiní odstranění nestvůry ze hry a skrytí soubojového okna, v případě že byl splněn úkol spustí přičtení zkušenostních bodů k aktuálním hodnotám bodů hrdinů a táže se frakce na to, jakou barvu úkolu si vyberou pro úkol, který splněná nahradí, a následně generuje nový úkol zvolené barvy.

Pokud přímá zranění nebyla dostatečná k zabití celé skupiny, zjišťuje hra, zda stačila k zabití alespoň části této skupiny. Pokud ano, dojde k zabití nejsilnější nestvůry, kterou je možno tímto počtem žetonů zabít.

Následně je vyhodnoceno zranění hrdinů. Pokud zbývající nestvůry útočí útokem v hodnotě vyšší než je počet všech žetonů (žetonů zranění i žetonů zbroje) v oblasti pro obranu, vypíše hra informaci o tom, kolik žetonů zranění si mají postavy účastnící se souboje odebrat.

Pokud nedošlo k zabití hrdinů, přesunou se žetony zranění ze sekce obrany a žetony oslabení do sekce útoku. Zde dojde k sečtení všech žetonů a vyhodnocení, zda jejich množství stačí k zabití skupiny nestvůr. Pokud ne, odstraní se žetony zbroje ze sekce obrany, ponechají se žetony v sekci útoku a následuje další kolo boje. Jinak je souboj vyhrán a proběhnou stejné akce jako při vítězství po první fázi boje.

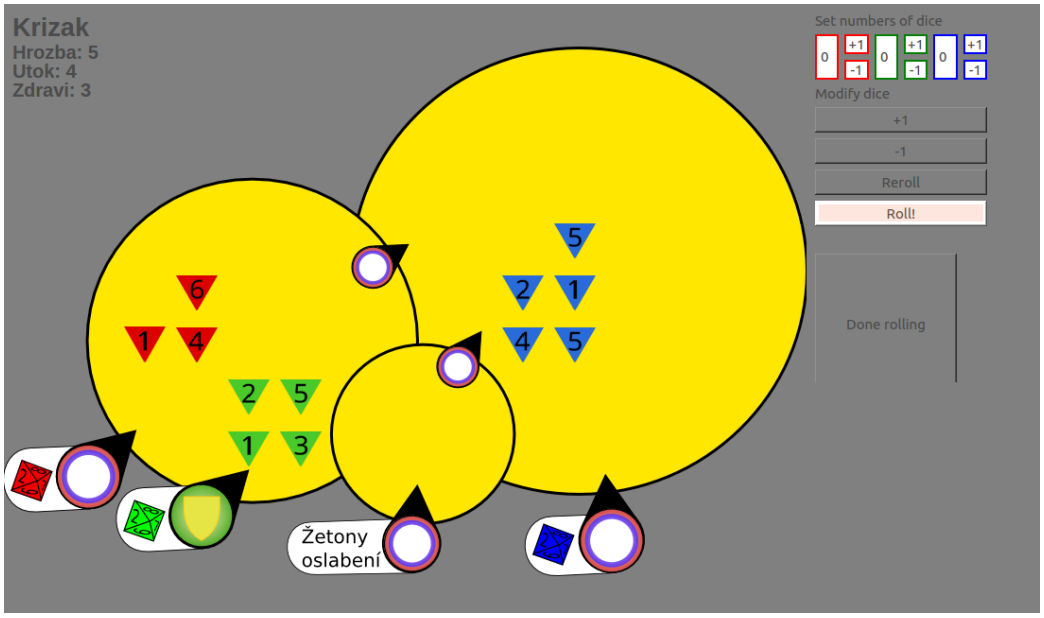
Vzhled soubojového okna je na obrázku 4.1.

4.4 Komplexní přehled tříd a jejich funkcionality

Pro snazší orientaci bych ráda uvedla přehled základních tříd, které se v aplikaci vyskytují, jaké funkce zajišťují, co je motivací k jejich existenci ve hře a poukázala na zajímavosti v kódu.

Gameboard

O této třídě nemluvím poprvé, již dříve jsem zmínila, že se tato třída stará o chod celé aplikace. Okamžitě v main funkci programu jsou totiž vytvořeny pouze dvě instance různých



Obrázek 4.1: Soubojové okno s hozenými kostkami.

tříd - třídy QApplication, která je potřebná kvůli využívanému Qt frameworku, a třídy Gameboard.

Snažila jsem se držet konceptu deskové hry, a jelikož desková hra je z definice hra taková, která se hraje na hracím plánu, hrací *desce*, snažila jsem se i třídu, přeložitelnou z angličtiny jako „hrací deska“, ponechat klíčovým prvkem aplikace.

Generuje veškeré startovní prvky, zajišťuje spuštění vláken pro ovládání aplikace, implementuje velkou část výpočtů hry (celý bojový systém, počítání zkušenostních bodů aj.). Tato třída dědí z Qt třídy QWidgets a veškeré ostatní prvky jsou jejími potomky, což umožňuje například aby Gameboard rozesílala přijímané zprávy o aktuálním doteku na dotykové ploše konkrétním správným prvkům, které se nachází na pozici doteku. Pro správu pozic svých potomků využívá jak různé typy třídy QLayout, tak absolutní pozicování na souřadnice, čímž můžu docílit přesného umístění prvků které se musí vztahovat například k pozici na obrázku, jedná se například o názvy regionů, které je nutno po mapě umístit na přesné souřadnice tak, aby jejich pozice odpovídala pozici kraje na mapě v pozadí.

Gameboard naslouchá na signály o stisknutí dotykové plochy `touched`, o kliknutí na hrdinu `characterClicked`, o boji s nestvůrami `fightSignal`, o změně viditelnosti prvků mapy a jiné. Veškeré signály zpracovává a většinu z nich pak vyšle do jiných tříd, které zajistí správnou reakci na tento signál.

MapRegion

Mapa hry je ve středu celého plánu a probíhají na ní veškeré akce. Mapa jako taková je ovšem jen obrázkem na pozadí.

Její regiony jsou však implementovány ve třídě MapRegion, která dědí z třídy QPushButton. Slouží především k udržování informací o prvcích v daném kraji (na základě poštu nestvůr a postav v kraji se přepočítává jejich umístění) a jejich tlačítka se využívají při cestování postavy. S tím souvisí i kontrola platnosti pohybu postavy.

Character

Třída Character zejména udržuje veškeré informace o postavě ve hře. Pro každou postavu existuje jedna instance této třídy. Jsou zde uchovávány informace o zkušenostních bodech, frakci, aktuálním regionu apod. Implementuje reakce na stisk postavy.

Quest

Třída Quest zaštiťuje aktuální úkoly. Udržuje veškeré statické informace o něm, stará se o vyvolávání skupin nestvůr na něm závislých, odměňuje hrdiny za splněné úkoly a náhodně generuje svá vlastní data z databáze úkolů pouze podle zadané barvy a frakce.

Tato třída je odvozena z Qt třídy QLabel. Třída QLabel, tudíž ani Quest, neumí přijímat zprávy z kliknutí myší (případně emulované kliknutí myší, které je vytvořeno při doteku). Veškeré pro hráče důležité informace o úkolu se tudíž zobrazují přímo.

Beast

Beast se stará o každou skupinu nestvůr na mapě. Udržuje její statické informace, které získala z databáze nestvůr. Třída Beast je také dědicem třídy QLabel, zobrazuje pouze základní informace o konkrétní nestvůře. Pozice vykreslování nestvůr jsou nastaveny absolutně do regionů mapy, ve kterých se nestvůry nachází.

Overlord

Jelikož má overlord zvláštní chování, je pro něj vyhrazena speciální třída. Při vytvoření instance této třídy se určí i typ overlorda, který následně ovlivňuje celé jeho chování. Tři typy overlordů totiž jsou Nefarian, který se pohybuje po mapě a přibližuje se do určitého kraje, kam když se dostane, vyhraje hru. Kel'Thuzad, který zůstává na jednom místě, ale přidává karty událostí s nepěknými efekty, a Lord Kazzak, který na mapě vytvoří sadu žetonů a schová se pouze pod jedním z nich, zbylé jsou falešné pozice, tudíž hráči ztratí čas cestováním a otáčením jednotlivých žetonů. Žetony jsou vyobrazeny na obrázku 4.2, celé informační panely o overlordovi na obrázku .

Obrázek 4.2: Žetony, kterými se overlordi zobrazují na mapě, zleva: Nefarian, Lord Kazzak, Kel'Thuzad, prázdná či neodkrytá pozice.

ActionCard

Akční karty, nebo také karty událostí, se otáčejí při určitých kolech. Spouští různé události, které mohou ovlivňovat průběh hry, nebo díky nim mohou hráči něco získat či uškodit druhé straně. Mohou také vyvolávat speciální typy nestvůr, takzvané bossy, za jejichž zabití zpravidla náleží postavám vysoká odměna, ale jejich zabití je náročnější než u běžných nestvůr. Další z typů základních karet jsou války, které se vždy odehrávají na určitém

Obrázek 4.3: Informační panely s detaily o overlordech.

území (např. „Válka o Stříbrobor“). Válku vyhrává ta frakce, která má jako první na konci kola kterékoliv frakce postavy ve dvou regionech, které jsou vyznačené na kartě. Na mapě se tyto regiony označí speciálními žetony války (viz obrázek 4.4).

Obrázek 4.4: Žetony označující válečné regiony na mapě.

Boss

Některé akční karty vyvolávají nestvůry, jež jsou úrovní náročnosti někde mezi klasickými nestvůrami a overlordem, těmto nestvůrám se říká bossové. Ve hře jich existuje pět v základních akčních kartách a další dva v akčních kartách overlorda Kel'Thuzada. Za jejich zabití náleží hráčům různá odměna, v původním plánu byla tedy jejich implementace jako speciální duh úkolu, nakonec jsem pro ně ale vyhradila celou novou třídu.

Dice

Speciální třídu mají vyhrazené také kostky. Původní návrh počítal s tím, že kostky budou zobrazeny pouze jako statické obrázky, narazila jsem ale na problémy s manipulací s již hozenými kostkami. Nakonec jsem tedy implementovala třídu Dice, která dědí z QPushButton. Každá kostka má svou hodnotu v rozsahu 1-8 a na základě této hodnoty se mění také její vzhled. Kostek je při spuštění hry vytvořeno 7 pro každou barvu a z průběhu hry se pouze nastavuje jejich viditelnost a hodnota.

4.5 Problémy při implementaci

Při implementaci jsem se setkala s nemálo problémy, z nichž některé bych zde chtěla zmínit. Většina z nich se týkala problémů s implementací reakce na doteky z dotykové vrstvy ARTable v C++.

Původní návrh počítal s implementací všech prvků mapy jako QWidget nebo QLabel (s funkcí pro odchyt událostí myši), kterým by se po stisku dotykové plochy zasílal funkcí `QApplicationCore::sendEvent` událost kliknutí myši. To se ale ani po extrémně dlouhé době nepodařilo zprovoznit, proto jsem se nakonec uchýlila k řešení pomocí třídy QPushButton, která implementuje funkci `click`. Toto řešení mi nepřijde natolik elegantní, ale povedlo se jej rychle implementovat a jeho funkčnost je dobrá.

Dále jsem se potýkala s problémem při odchytu dotyku na souřadnicích. Problémy se vyskytly prakticky ihned při odchytu prvních doteků, kdy souřadnice nedávaly vzhledem k aplikaci systém. Experimentálně jsem pak zjistila, že zatímco Qt aplikace má začátek souřadného systému v levém horním rohu, dotyková plocha za nulový bod považuje roh levý dolní. Tento problém se naštěstí dal velmi jednoduše opravit přepočítáním souřadnic při odchytu.

Když už ale souřadnice byly správné, stále jsem nedokázala posílat události do správného prvku. Funkce mi totiž vždy vracela první prvek, který na daných pozicích byl, což velmi často znamenalo QWidget obsahující celou mapu, přinejhorším i celou Gameboard. To jsem nakonec vyřešila vyhledáváním v potomcích třídy Gameboard, kteří jsou zároveň tlačítkem, a následným porovnáním jejich souřadnic, výšky a šířky se souřadnicemi stisku. Tlačítko se tak zmáčkne pouze v případě, že byl stisk proveden v oblasti tlačítka a pokud je tlačítko viditelné. Pokud některá z vlastností neodpovídá, testuje se další tlačítko, dokud se neprojde celý list potomků. O efektivitě tohoto řešení by se dalo polemizovat, bohužel jsem nenašla efektivnější způsob.

4.6 Testování

Testování proběhlo v laboratoři se stolem ARTable. Nejprve jsem požádala postupně různé kamarády, kteří se na hru přišli podívat, osahat si ji a sdělili mi svou kritiku.

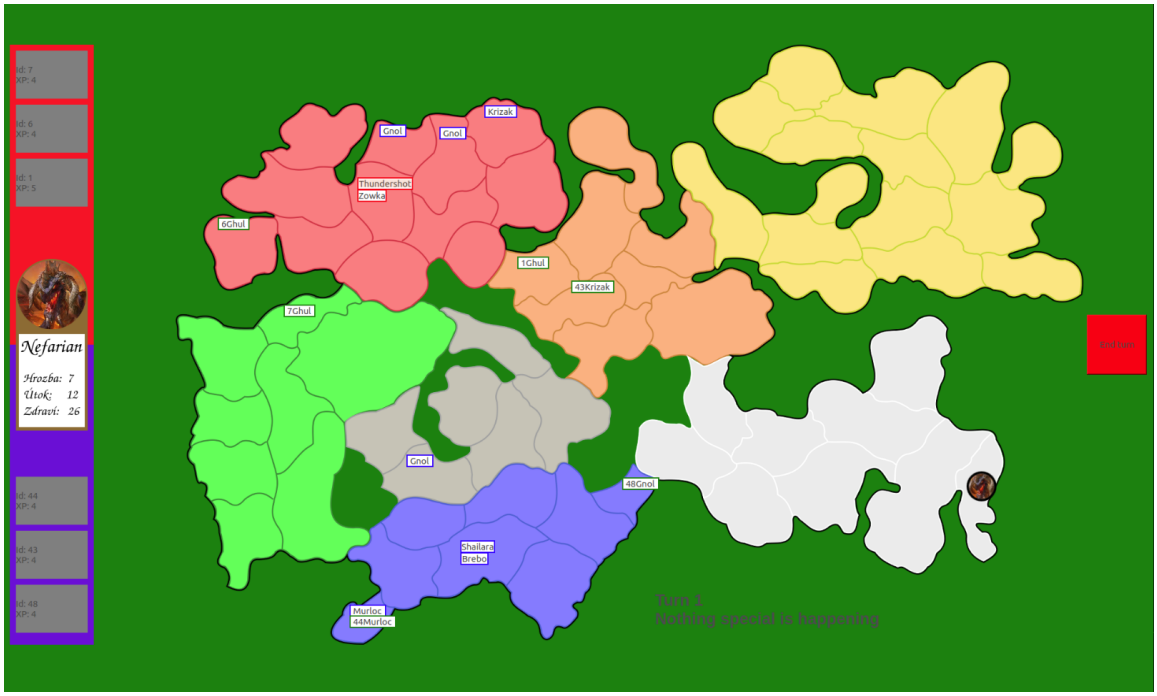
Na závěrečné testování jsem pozvala 4 hráče s různými úrovněmi znalostí originální hry a vyzvala je ke hře bez delšího vysvětlování. Obdrželi veškeré fyzické prvky, které jsou nutné ke hře, a bylo jim vysvětleno které prvky se zobrazí na hracím plánu virtuálně.

Začátek hry nebyl zcela plynulý, hráči často registrovali nereagování plochy na dotek a vadila jim občas delší prodleva mezi dotykem a zobrazením změny na plánu. Dále jim vadilo, že nešlo v této době prodlevy poznat, zda bylo tlačítko stisknuto nebo ne, takže často docházelo k tomu, že svůj dotek zopakovali, což zapříčinilo spuštění akce dvakrát (někdy to příliš nevadilo, ale jindy tak třeba hráči jedné frakce zcela zabránili tahu druhé frakce při dvojím kliknutí na tlačítko ukončení tahu).

Hráči byli také nespokojení s rozlišením promítaného obrazu a některé texty se pak, díky rozlišení a špatně zvolené velikosti a barvě, špatně četly.

Na základě byly upraveny velikosti tlačítek, některé barvy a velikosti textů. Zasílání obrazu na projektor bylo nastaveno na pětinu původního času, aby hra reagovala na doteky v rychlejším čase.

Výsledný vzhled hry je zobrazen na obrázcích 4.5 a 4.6.



Obrázek 4.5: Výsledné grafické uživatelské rozhraní pro hru World of Warcraft Board Game

Obrázek 4.6: Výsledné grafické uživatelské rozhraní pro hru World of Warcraft Board Game zobrazené na ARTable.

Kapitola 5

Závěr

Tato bakalářská práce se věnovala tvorbě promítaného uživatelského rozhraní v rozšířené realitě. Jejím výsledkem je hratelná verze hry World of Warcraft Desková hra na platformě ARTable. Pro tuto práci bylo implementováno promítání aplikace a její ovládání přes dotykovou vrstvu.

Práce mě přiměla nastudovat materiály z různých oblastí, od tvorby uživatelských rozhraní v jazyce C++ s využitím technologie Qt, přes databáze v úsporném SQLite až k rozšířené realitě pomocí frameworku Robot Operation System.

Z teoretické stránky byla představena témata rozšířené reality, promítaných uživatelských rozhraní, způsobů zobrazování a ovládání promítaného uživatelského rozhraní, v krátkosti také platforma ARTable. Podrobněji byl pak předveden framework Robot Operation System, jehož ovládnutí bylo k této práci esenciální.

Hře bohužel chybí některé minoritní prvky, které by mohly hráčům ještě usnadnit či zpříjemnit hraní, proto stále vidím možnosti kam se posouvat.

Ráda bych v budoucnosti rozšířila tuto hru o ovládání pomocí gest a snímání značek skrze Kinecty a zároveň přispěla novými ukázkovými aplikacemi k ARTable pro jazyk C++, neboť nyní jsou všechny psané pro jazyk Python.

Literatura

[1] About ROS. [Online] [cit. 2018-05-06].
URL http://www.ros.org/about-ros/

[2] ARTable na VUT FIT. [Online] [cit. 2018-05-11].
URL http://www.fortes.cz/portfolio/artable-na-vut-fit-2/

[3] Ubuntu install of ROS Indigo. [Online] [cit. 2018-05-11].
URL http://wiki.ros.org/indigo/Installation/Ubuntu

[4] Azuma, R. T.: A Survey of Augmented Reality. *Presence: Teleoperators and Virtual Environments*, 1997, ISSN 1531-3263.

[5] Bimber, O.; Emmerling, A.; Klemmer, T.: Embedded entertainment with smart projectors. *Computer*, ročník 38, 2005: s. 48 – 55, ISSN 0018-9162.

[6] Bimber, O.; Raskar, R.: *Spatial Augmented Reality*. A K Peters, Ltd., 2005, ISBN 1-56881-230-2.

[7] Birkfellner, W.; Figl, M.; Huber, K.; aj.: A head-mounted operating binocular for augmented reality visualization in medicine - design and initial evaluation. In *IEEE Transactions on Medical Imaging*, ročník 21, IEEE, Srpen 2002, ISSN 0278-0062.

[8] Jordà, S.: The reactable: tangible and tabletop music performance. In *CHI'10 Extended Abstracts on Human Factors in Computing Systems*, ACM, 2010, s. 2989–2994.

[9] Lee, D.: Capacitive vs. Resistive Touchscreens. [Online] [cit. 2018-05-11].
URL http://articles.r-tt.com/gadgets/tablets/capacitive-resistive-touchscreens

[10] Müller-Tomfelde, C.: *Tabletops-Horizontal Interactive Displays*. Springer Science & Business Media, 2010, ISBN 978-1-84996-112-7.

[11] Pitts, M. J.; Skrypchuk, L.; Attridge, A.; aj.: Comparing the User Experience of Touchscreen Technologies in an Automotive Application. In *AutomotiveUI '14*, 2014, ISBN 978-1-4503-3212-5.

[12] Quigley, M.; Gerkey, B.; Conley, K.; aj.: *ROS: an open-source Robot Operating System*. Willow Garage.

[13] Steinberg, G.: Natural User Interfaces. In *CHI '12 Extended Abstracts on Human Factors in Computing Systems*, 2012, ISBN 978-1-4503-1016-1.

Příloha A

Obsah přiloženého paměťového média

Součástí této práce je přiložené CD s následující strukturou

- *xheles02_BP.pdf* - tato bakalářská práce ve formátu pdf,
- *xheles02_plakat.pdf* - plakát vytvořený pro ilustraci práce,
- *xheles02* - složka se strukturou catkin balíčku, která obsahuje zdrojové kódy a jiné soubory nutné k přeložení a správnému spuštění a běhu aplikace,
- *xheles02_BP_tex* - složka se zdrojovými soubory k technické zprávě.

Příloha B

Konfigurace systému

Jelikož je ARTable velmi specifický systém, je potřeba před pokusem o spuštění samotného programu hry nakonfigurovat prostředí, kde se bude hra spouštět. To vyžaduje instalaci několika různých prostředí a balíčků. K popisu správné konfigurace slouží tato příloha.

Verze frameworku ROS Indigo, jež je používána v této práci, vyžaduje využití operačního systému Ubuntu 13.10 Saucy nebo Ubuntu 14.04 Trusty. Instalace samotného ROS frameworku je pak poměrně jednoduchá.

Před zahájením instalace rosu doporučuji první nainstalovat Python, pokud již v systému není. Na svém systému pracuji s Pythonem ve verzi 2.7.6.

Prvním krokem k instalaci je pak samozřejmě aktualizace systému. Instalace samotná se poté spouští příkazem `sudo apt-get install ros-indigo-desktop-full`. Před tím, než bude ROS možno použít, je nutné inicializovat `rosdep`, který se stará o instalaci některých závislostí a je nutný ke spuštění některých komponent rosu. Dále je nutné nastavit některé proměnné prostředí a nainstalovat `rosinstall`, což je separátně dodávaný nástroj, který umožňuje snadnou instalaci zdrojů pro balíčky rosu.[3]

K ovládání ARTable je dále potřeba nainstalovat `libfreenect2` z repozitáře na adrese https://github.com/OpenKinect/libfreenect2 a `ial_kinect2` z repozitáře na adrese https://github.com/code-iai/iai_kinect2. Tyto dva repozitáře zajišťují podporu používání Kinectu. Součástí obou repozitářů je přesný návod k instalaci.

Jelikož ARTable využívá build systému Catkin, je nutné vytvořit catkin workspace sadou příkazů

```
mkdir -p catkin_ws/src
cd catkin_ws/
catkin_make
```

.
Příkaz catkin_make vytvoří kompletní prostředí pro catkin.

Poté už zbývá pouze repozitář samotného ARTable. Ten se nachází na adrese https://github.com/robofit/artable. Jeho klon musí být umístěn ve složce /catkin_ws/src. Pokud byla celá příprava systému úspěšná, dá se nyní repozitář ARTable sestavit příkazem `catkin_make`.

Balíček hry pak můžete zkopírovat z přiloženého CD, přeložit příkazem `catkin_make` a spustit pomocí `rosrun xheles02 xheles02`.